Pascal (langage)

Pascal est un langage de programmation impératif qui, conçu pour l'enseignement, se caractérise par une syntaxe claire, rigoureuse et facilitant la structuration des programmes[1].

En dehors de la syntaxe et de sa rigueur, le langage Pascal possède des points communs avec le C (voir les pointeurs). Le langage Pascal de base était conçu à usage purement éducatif et était assez limité. Par exemple, les chaînes de caractères, absentes du langage d'origine, ont rapidement été intégrées[2, 3].

Sa puissance a été établie, et sa diffusion rapidement facilitée par la rédaction de compilateurs Pascal écrits en Pascal.

Les développements qu'il a connus par la suite en ont fait un langage complet et efficace. Plus récemment, la généricité a été ajoutée dans Delphi 2009[4] et dans Free Pascal depuis la version 2.2[5, 6].

Les implémentations actuelles de Pascal, utilisées hors du monde éducatif, sont des extensions telles que Turbo Pascal (mode texte), Pascal Objet (programmation objet), et Delphi (fenêtré). Il existe des versions libres comme Free Pascal et Lazarus (fenêtré). On peut programmer en Pascal sous DOS, Windows, Mac OS ou encore sous Linux/Unix ou Palm OS.

Le système d'exploitation des ordinateurs Apollo[7], ainsi qu'une partie du système du Macintosh ont été écrits en Pascal. La première version d'Adobe Photoshop également[8]. Le compilateur GCC a été développé par Richard Stallman à partir d'un compilateur du LLNL, qui était écrit en langage Pastel, une extension du langage Pascal[9].

La syntaxe du langage a été adaptée à d'autres langages comme Ada, Modula-2 (puis Modula-3) ou Oberon.

Pascal	
Date de première version	1970
Paradigme	générique, orientée objet, procédural, impératif
Auteur	Niklaus Wirth
Typage	statique, sûr, nominatif
Dialectes	ISO Pascal, UCSD Pascal, Turbo Pascal, Apple Pascal, Delphi, Free Pascal
Influencé par	Algol, Fortran
A influencé	Ada, Modula-2, Modula-3, Oberon
Implémentations	CDC 6000, PASCAL-P, PDP-11, PDP-10, IBM System/370, HP, GNU Pascal, Delphi, Free Pascal, Lazarus
Extension de fichier	pas

Sommaire

Présentation et histoire

Le langage de programmation Pascal (dont le nom vient du mathématicien français Blaise Pascal) a été inventé par Niklaus Wirth dans les années 1970 avec l'aide d'un de ses étudiants, Urs Amman. Il a été conçu pour servir à l'enseignement de la programmation de manière rigoureuse mais simple, en réaction à la complexité d'Algol 68. Le premier compilateur a été conçu sur un CDC 6400[1, 10].

Ce langage est l'un de ceux qui ont servi à enseigner la programmation structurée. Le *goto* ou saut n'importe où dans le programme (dit « branchement ») est fortement déconseillé dans ce langage, le programme est un assemblage de procédures et de fonctions, dans lesquelles on peut utiliser des blocs conditionnels (*if*, *case*) et répétitifs (*while*, *for*, *repeat*) ayant chacun une entrée et une sortie afin de faciliter les contrôles, ce qui aboutit à des mises au point rapides et sûres.

Le langage est de plus fortement et statiquement typé, c'est-à-dire que toutes les variables doivent avoir un type défini au moment de la compilation. En revanche son manque de souplesse pour gérer les passages du type caractère au type chaîne de caractères est l'un de ses points faibles.

Il a largement pénétré le monde de l'éducation et de la recherche (universités), puis dans une moindre mesure celui de l'industrie et de l'édition logicielle.

Le compilateur P4 a été diffusé en source dans les universités, à un prix très réduit. Il générait du P-Code, un code pour une machine virtuelle. Les programmes Pascal étaient donc facilement portables sur une machine. Il suffisait d'écrire pour elle un interpréteur de P-Code. Il y eut donc rapidement des portages sur DEC PDP-11, premier support de C, et sur 6502, 8080 et Z80, les principaux microprocesseurs de l'époque.

Le compilateur UCSD Pascal, de l'université de Californie à San Diego, eut beaucoup de succès, notamment sur des machines comme l'Apple II qui furent très diffusées[11].

Mais le coup de « turbo » sera donné par la société Borland, créée en 1983 qui commercialisa le compilateur Turbo Pascal pour un prix très modique (49 $ de l'époque alors que le compilateur Pascal Microsoft était à plus de 500 $ et ne possédait pas d'éditeur intégré ni de compilation *in core*). Quick Pascal de Microsoft sera commercialisé bien trop tard pour inverser la tendance.

Des compilateurs ont été produits pour divers ordinateurs, notamment des fabricants Sun[12], HP[13], SGI[14], CDC[15], IBM[16], Unisys[17], Texas Instruments[18].

Le Pascal a fait l'objet des normes ISO 7185 (1983)[19] et ISO 10206 (1990)[20].

Les fichiers sources

En général, on reconnaît un programme en Pascal par l'extension du fichier qui le contient : *.pas*, *.p* ou *.pp*, les deux dernières étant plus fréquentes sur les systèmes de type UNIX. L'utilisation d'environnements fenêtrés (Delphi et Lazarus) a entrainé l'apparition de nouvelles extensions.

Le code source est organisé suivant différentes possibilités :

- Un programme principal commence par le mot-clé *program*, facultatif dans certaines implémentations. Il est compilé et lié avec les unités qu'il utilise en un fichier exécutable. Cette phase de compilation est le cas le plus fréquent dans les compilateurs récents. Au contraire, le Pascal UCSD, par exemple, produisait du bytecode.
- Une bibliothèque dynamique commence par le mot-clé *library*. Elle est compilée en un fichier portant l'extension .dll sous Windows, .so sous UNIX. Les bibliothèques dynamiques sont apparues dans le langage Pascal en même temps que Delphi.
- Un fichier unité commence par le mot-clé *unit*. Il est inclus dans des programmes ou des bibliothèques, et peut être compilé en un code objet standard (extension .o ou .obj) ou dans un format particulier, tel que *.dcu* pour Delphi et .ppu pour Lazarus.

Avant l'apparition des bibliothèques, Turbo Pascal permettait d'utiliser des *Overlay* **(en)**, technique habituelle sous DOS pour les programmes de grande taille[21]. Il s'agissait de fichiers séparés du fichier exécutable principal et qui pouvaient être chargés ponctuellement.

Il est possible d'inclure du code dans un programme autrement qu'en écrivant une unité et ce en faisant simplement un *include*, c'est-à-dire en indiquant au compilateur d'inclure le texte d'un fichier dans un programme avec la directive *$I* :

```
program test;

uses Crt, Dos; { utilisation habituelle des unités }

{$I outil.inc} { utilisation directe de code externe }

begin
    { utiliser les fonctions du fichier outil.inc }
end.
```

Toutefois, cette méthode n'est pas recommandée au regard des inconvénients qu'elle présente - notamment si un identificateur est déclaré plusieurs fois dans des fichiers différents - et du manque de contrôle du compilateur sur ce type de fichier.

Le Turbo Pascal

Le logiciel Turbo Pascal a été écrit par Anders Hejlsberg[22, 23] : il s'est appelé auparavant Compass Pascal, puis Poly Pascal. Très compact (12 kilooctets) et très rapide car travaillant essentiellement en mémoire vive, il compilait en une passe et produisait du code machine x86 sous DOS et non pas du bytecode. Il était livré avec un environnement complet (un éditeur de texte et une aide en ligne, innovation à l'époque, particulièrement compacte grâce à un système de *substitution*).

Chaque nouvelle version de Turbo Pascal a apporté son lot d'innovations, comblant en partie des lacunes du langage original. C'est ainsi qu'en 1987 la version 4 apparaît avec un véritable environnement de développement intégré, en 1989 la version 5.5 introduit les *objets*[24], en 1990 la version 6 permet la programmation de fenêtres dans la console DOS, par le biais de la Turbo Vision, apportant

les prémisses de la programmation événementielle. Enfin, en 1992 sort la version 7 pour DOS, qui sera la dernière, ainsi qu'une version pour Windows, rapidement supplantée par Delphi.

Un exemple de code : Hello World

```
program HelloWorld(output);
begin
  writeln('Hello World');
  readln;
end.
```

Le paramètre `Output` qui suit le nom du programme est aujourd'hui facultatif (il n'était obligatoire qu'avec les premières versions des implémentations Pascal). De nos jours, il est absent la plupart du temps.

Contrairement au C, le Pascal n'est pas sensible à la casse, c'est-à-dire que les mots réservés (comme `begin`) ou les identificateurs (comme `write` ou la variable `a`) peuvent être indifféremment écrits en majuscules ou en minuscules.

Toujours contrairement au C, les déclarations se font dans une partie clairement séparée du code. Les déclarations locales sont faites en début de procédure ou de fonction, les déclarations globales, elles, étant faites n'importe où avant le programme principal. Ceci ajoute de la clarté au langage au prix d'un certain manque de souplesse. On ne peut pas déclarer de variable au beau milieu d'une fonction. Notons qu'en Pascal, les déclarations doivent précéder toute utilisation ; il est notamment interdit d'utiliser une procédure ou une fonction qui n'a pas encore été déclarée. Il est possible de contourner ce problème à l'aide du mot-clé *forward*.

Enfin, la distinction entre procédures et fonctions permet d'éviter certaines erreurs de programmation - par défaut, car une directive de compilation de Delphi permet d'activer une syntaxe « étendue » qui offre le même laxisme que le C.

Delphi et la programmation fenêtrée

En 1995, pour contrecarrer Microsoft et la programmation visuelle du Visual Basic, Borland sort Delphi. Contrairement à VB qui produit du p-code, Delphi produit du code machine, plus rapide. On voit également apparaître la bibliothèque VCL servant d'interface aux bibliothèques système **(en)** de Windows, facilitant grandement le développement.

Au début des années 2000, Borland produit Kylix, l'équivalent de Delphi pour le monde Linux, qui n'aura pas un grand succès. Au début des années 2010, Embarcadero, qui a repris les activités d'outils de développement de Borland, produit Delphi XE, dont les versions récentes sont compatibles avec Windows, OS X et iOS.

Lazarus, est un logiciel libre de développement intégré RAD légèrement différent, permettant de compiler sur différents systèmes d'exploitation tel que Windows, GNU/Linux, OS X, Unix, OS/2, ReactOS, Haiku et plateformes telles que x86, x86-64, ARM, SPARC, PowerPC, IA-64.

Exemple de source Lazarus

Voici un exemple de fichier Pascal associé à une fenêtre (`Form1: TForm1`) contenant un label (un texte). Ce fichier est généré automatiquement et sert de structure de base pour programmer. C'est-à-dire qu'on peut le modifier, le compléter etc. Les petits points sont des repères lors de l'édition. Ils sont invisibles à l'exécution. Le source et les fenêtres Delphi sont très semblables.


```
unit Unit1;

    {$mode objfpc}{$H+}

interface

uses
    Classes, SysUtils, LResources, Forms, Controls, Graphics,
```

```
Dialogs, StdCtrls;

type
    { TForm1 }

    TForm1 = class (TForm)
         Label1: TLabel; { le label "Hello world!" posé sur la
fenêtre }
    private
         { private declarations }
    public
         { public declarations }
    end;

var
    Form1: TForm1;

implementation

initialization
    {$I unit1.lrs}

end.
```

Petite explication : la directive *{$I unit1.lrs}* permet de lier la classe TForm1, décrivant une fenêtre, au fichier de ressource *unit1.lrs* qui contient le design de la fenêtre. Avec Lazarus, le fichier *lrs* est un fichier intermédiaire créé automatiquement par le compilateur à partir des informations du fichier *lfm* et des directives de compilation (notamment la possibilité de choisir la bibliothèque de widget). Avec Delphi, la directive équivalente aurait été *{$R unit1.dfm}* et il n'y a pas de fichier intermédiaire. Par ailleurs, elle aurait été placée dans la partie interface.

Compilateurs

Parmi les compilateurs encore utilisés aujourd'hui (2013), on peut citer :

- Free Pascal en particulier accompagné du compilateur et de l'environnement de développement libre RAD Lazarus, vise à la meilleure compatibilité possible avec Delphi. Il existe sur plusieurs plateformes, facilitant le portage d'un programme d'un environnement à un autre.
- Delphi, compilateur et environnement de développement « RAD » commercial. Les versions récentes de Delphi fonctionnent sur Windows et MacOS X. Au début des années 2000, Borland a tenté une version pour Linux, Kylix, qui n'a pas eu le succès escompté. Kylix utilisait la CLX, framework ressemblant à la VCL, mais utilisable sur les deux systèmes d'exploitation. Dans Delphi XE2 Embarcadero a introduit le même principe de compatibilité entre systèmes, avec FireMonkey **(en)**
- GNU Pascal, dont le but est la compatibilité avec la norme ISO : il implémente complètement l'ISO 7185, et en grande partie l'ISO 10206. Son développement n'est plus actif depuis 2006.

Aspects techniques

Récursivité

On peut écrire des fonctions récursives en Pascal. La fonction suivante en donne un exemple :

```
function factorielle(n: integer): integer;
begin
  if n <= 1 then
    factorielle := 1
  else
    factorielle := n*factorielle(n - 1);
end;
```

Le nombre d'appels récursifs est limité par la pile. Certaines implémentations sont toutefois capables d'optimiser la récursion terminale, par exemple Free Pascal avec l'option *-O2*.

Généricité

Il est possible de définir des types génériques, c'est-à-dire pouvant être utilisés avec différents types possibles sans pour autant avoir à réécrire le code. Voilà un exemple en Free Pascal :

```
program Pj_gen;
uses SysUtils;
type
  { On définit un objet qui permet d'additionner deux éléments et multiplier un élément par un entier
strictement positif }

  { TAnneauGenerique }
  generic TAnneauGenerique<T> = class
    function Somme(a, b: T): T;
    function Multiple(n: integer; a: T): T;
  end;

  { On crée deux versions spécialisées, s'appliquant aux nombre réels et aux chaines de caractères }
  TAnneauReel = specialize TAnneauGenerique<Real>;
  TAnneauChaine = specialize TAnneauGenerique<String>;

{ TAnneauGenerique }

function TAnneauGenerique.Somme(a, b: T): T;
begin
  result := a + b;
end;

function TAnneauGenerique.Multiple(n: integer; a: T): T;
var i: integer;
begin
     { On ne gère que le cas où n >= 1 }
     result := a;
     for i := 2 to n do result := result + a;
end;

var
  r: TAnneauReel;
  c: TAnneauChaine;

begin
  r := TAnneauReel.Create;
  writeln('Avec des reels :');
  writeln('4.1 + 5.2 = ', FloatToStr( r.Somme(4.1, 5.2) ));
  writeln('3 x 5.1 = ', FloatToStr( r.Multiple(3, 5.1) ));
  r.free;

  writeln;

  c := TAnneauChaine.Create;
  writeln('Avec des chaines :');
  writeln('bon + jour = ', c.Somme('bon', 'jour'));
  writeln('10 x a = ', c.Multiple(10, 'a'));
  c.free;

  writeln;
  writeln('Press <Enter> to quit');
  readln;
end.
```

Résultat sous FPC2.6.0:

```
Avec des reels :
4.1 + 5.2 = 9,3
3 x 5.1 = 15,3

Avec des chaines :
bon + jour = bonjour
10 x a = aaaaaaaaaa

Press <Enter> to quit
```

Note : en Delphi, il ne faut pas ajouter les mots-clés `generic` et `specialize`.

Dépendances mutuelles des unités

Il est possible que les différentes unités du programme s'utilisent mutuellement, mais il y a une limitation. En effet, en Pascal, on peut faire référence à une autre unité à deux endroits possibles, à savoir dans l'interface et à l'implémentation. Si deux unités font référence l'une à l'autre dans leur interface, on obtient une erreur de dépendance circulaire. On peut contourner ce problème en utilisant une troisième unité qui servira de base aux deux autres, et/ou en utilisant des types plus abstraits au niveau des interfaces. Cette limitation n'existe pas en Visual Basic.

Critique du langage

En 1981, Brian Kernighan et Phillip J. Plauger **(en)** publient le livre *Software Tools in Pascal*, réédition de leur ouvrage précédent, *Software Tools*, publié en 1976, et qui employait le langage Rational Fortran **(en)**. Le but était de fournir, en langage Pascal, des programmes complets et utiles[25], bien documentés, et montrant comment écrire de « bons » programmes. À noter que Niklaus Wirth avait publié en 1979 une collection de programmes visant un objectif similaire[26]. Les *Software Tools* étaient écrits dans le langage défini par l'ouvrage de Kathleen Jensen et Niklaus Wirth, *Pascal User Manual and Report* de 1978, et par la proposition de standard ISO. La même année, Brian Kernighan publia l'article *Why Pascal is not my Favourite Language*[27], dans lequel il dénonçait les défauts qu'il voyait dans le langage, et qui selon lui empêchaient de l'utiliser pour de la « programmation sérieuse ». L'article partait de son expérience avec l'ouvrage précédent, et de la comparaison qu'il avait pu faire avec le C, dont il assurait par ailleurs la promotion - le livre *The C Programming Language*, coécrit avec Dennis Ritchie, était sorti en 1978[28].

Parmi les aspects contestés dans cet article, l'un d'eux rendait la programmation en Pascal particulièrement compliquée : le typage des tableaux, et par voie de conséquence, le typage des chaînes de caractères également. En effet, les dimensions des tableaux font partie du type, en Pascal, ce qui empêche de passer à une fonction des tableaux de taille variable (ou des chaînes de taille variable). Contourner le problème oblige soit à écrire de multiples versions des fonctions qui prennent des tableaux en paramètres, soit à utiliser un type tableau de la taille maximum estimée. Cette dernière « astuce » était fréquemment utilisée dans ce langage ; c'est ainsi que les programmes des *Numerical Recipes in Pascal* y font systématiquement appel. Cet inconvénient, toujours présent dans la version définitive du langage ISO 7185 de 1983, était bien connu dans le milieu de l'analyse numérique. Ainsi, lors de la *Conference on the Programming Environment for Development of Numerical Software*[29], organisée en 1978 par le Jet Propulsion Laboratory et l'ACM SIGNUM[30], une des présentations[31] montrait l'utilisation possible du Pascal en analyse numérique, et pointait ce problème de passage de tableau, en proposant une syntaxe alternative destinée à être intégrée à une version ultérieure du standard. De fait, elle ser ajoutée en 1990 à l'ISO 10206 *Extended Pascal*. Ce dernier permet, via les *schémas*, de créer des types structurés de taille dynamique. De même en Extended Pascal, tous les types chaînes de caractères sont compatibles entre eux[32]. Mais peu de compilateurs ont suivi le standard ISO 10206 à la lettre, en particulier ni Turbo Pascal ni les compilateurs dérivés de celui-ci (Delphi et Free Pascal).

Évolution de Pascal

Aujourd'hui, ce problème n'est plus d'actualité, puisque les versions récentes de Delphi et Free Pascal permettent plusieurs alternatives :

- l'allocation dynamique avec *getmem* et *freemem*, puis un accès au « tableau » par le biais d'un pointeur. C'est l'exact équivalent de ce qui est fait en C,
- les tableaux dynamiques, avec le dimensionnement par *SetLength*[33],
- les « tableaux ouverts », des paramètres de fonction dont la dimension n'est pas précisée[34] : c'est ce qui se rapproche le plus de ce que permet le standard ISO.

Par ailleurs, une autre solution, bien plus pertinente en Pascal Objet, consiste à définir une classe encapsulant les matrices et leurs opérations. Les arguments passés aux fonctions sont alors des objets.

Quant aux chaînes de caractères, Delphi et Free Pascal fournissent le type *AnsiString*[35], dont la dimension n'est pas limitée, et plus important pour ce qui nous préoccupe ici, ne fait pas partie du type. En réalité, ces chaînes sont des pointeurs dont la gestion est faite de façon transparente pour l'utilisateur.

Une autre critique, pointée par les *Numerical Recipes in Pascal*, concerne le passage de fonction comme paramètre, très utile en analyse numérique. C'est indispensable pour écrire une bibliothèque de calcul la plus générale possible : ainsi, pour du calcul intégral, on écrit une fonction réalisant le calcul dans le cas général, la fonction à intégrer étant inconnue, et on la lui passe en paramètre dans les cas particuliers dont on a besoin. Cette fonctionnalité fait elle aussi partie du Pascal étendu, et n'a pas été utilisée dans l'ouvrage précédemment cité, car peu de compilateurs l'implémentaient alors. Là encore, ce n'est plus un problème avec Delphi et Free Pascal, qui acceptent les « types procéduraux »[36], pouvant être passés en paramètres.

Descendance et Parenté

Au-delà des variations commerciales, un certain nombre d'utilisateurs Pascal soucieux de fiabilité sont passés soit au langage Portal, soit au langage Modula 2 et ses descendants, soit aux langages apparentés comme Ada 83, ou enfin aux langages objets comme Simula, Modula 3, Oberon ou Ada 95.

En particulier, Modula comme C suffit à écrire un système d'exploitation, mais avec une plus grande sécurité du fait d'une séparation claire entre le niveau physique - le plus délicat dans les opérations de portage - et le niveau symbolique, le plus facile à contrôler et le plus employé.[pas clair]

Pascal dans l'enseignement

En France

Le portage par l'INRIA du Pascal/CDC sur l'Iris 80 a mis ce langage à la disposition des centres de calcul universitaires à la fin des années 1970. À partir de 1980, des universités et des écoles d'ingénieurs s'en sont servies comme support de l'algorithmique et de la programmation structurée. En effet, il était :

- plus efficace et plus ouvert qu'Algol 60,
- plus ouvert, plus structuré et moins permissif que Fortran,
- plus clair, plus structuré et moins permissif que PL/I.

L'arrivée des micro-ordinateurs introduisit une certaine compétition entre Pascal UCSD, Microsoft Pascal, et Turbo Pascal.

L'enseignement de l'informatique en classes préparatoires n'est introduit qu'en 1987, basé sur Pascal (plus précisément Turbo Pascal). Une bibliothèque (« MODULOG »), développée par l'ALESUP[37] et l'IREM de Marseille, était également mise à disposition des lycées[38, 39, 40].

Avec la réforme de 1995, l'enseignement de l'informatique en prépa scientifique est séparé en un tronc commun, basé sur un logiciel de calcul formel, et une matière optionnelle en MPSI et MP basée, au choix, sur Pascal ou Caml. Le succès de ce dernier est tel qu'au « stage de Luminy » en 1997, seulement trois participants choisissent le Pascal[41]. Très minoritaire dans la filière MP, Pascal est officiellement abandonné à partir de la session 2015 des concours, et depuis seul Caml est autorisé aux épreuves de l'option informatique.

Les prépas BCPST ont suivi un chemin similaire : le Pascal est introduit au programme en 1987, puis retiré au début des années 2000, au profit de MATLAB et Scilab. Le Pascal n'est plus proposé aux concours à partir de 2004[42].

En Suisse

Pascal a d'abord été largement adopté. Par la suite, les travaux pratiques et développements combinant Pascal et assembleur ont amené à le remplacer par le langage Portal.

Bibliographie

- Niklaus Wirth, Kathleen Jensen, *Pascal User Manual and Report: ISO Pascal Standard*, 4e éd., Springer, 1991 (ISBN 978-0387976495)
- Niklaus Wirth, *Introduction à la programmation systématique*, trad. par Olivier Lecarme / Masson, Paris, 1977
- Sanford Leestma, Larry Nyhoff, *Pascal Programming and Problem Solving*, 4e éd., Prentice Hall, 1993 (ISBN 978-0023887314)
- Nell Dale, *Programming in Pascal*, Jones and Bartlett, 1997 (ISBN 978-0763704841)

Voir aussi

- Lazarus
- Java
- MIDletPascal
- Virtual Pascal

Sur les autres projets Wikimedia :

Pascal (langage), sur Wikiversity

la programmation en Pascal, sur Wikibooks

Liens externes

- **(fr)** Livres de Patrick Cousot sur la programmation en Pascal
- **(fr)** Cours et tutoriels sur Pascal sur Developpez.com

Notes et références

1. Summary of projects by N. Wirth, 1962 - 1999 (http://www.inf.ethz.ch/personal/wirth/projects.html)
2. The Programming Language Pascal (Revised Report) (ftp://ftp.inf.ethz.ch/pub/publications/tech-reports/1xx/005.pdf) Niklaus Wirth, ETH Zurich, 1973
3. An Axiomatic Definition of the Programming Language Pascal (ftp://ftp.inf.ethz.ch/pub/publications/tech-reports/1xx/006.pdf), C. A. R. Hoare, Niklaus Wirth, ETH Zurich, 1972
4. Delphi 2009 Reviewers Guide (http://edn.embarcadero.com/article/38757)
5. Generics - Free Pascal wiki (http://wiki.freepascal.org/Generics)
6. Generics (http://www.freepascal.org/docs-html/ref/refch8.html) dans le manuel de référence de Free Pascal
7. Documentation des ordinateurs Apollo (http://bitsavers.trailing-edge.com/pdf/apollo/)
8. Computer History Museum - Adobe Photoshop Source Code (http://www.computerhistory.org/atchm/adobe-photoshop-source-code/)
9. A Brief History of GCC (http://gcc.gnu.org/wiki/History)
10. The CDC 6000 Series Computer (http://www.standardpascal.org/cdc6400.html)
11. Documentation pour la programmation de l'Apple II GS (http://www.apple-iigs.info/docprogrammation.php)
12. Documentation du Pascal de Sun (http://docs.oracle.com/cd/E19957-01/)
13. Documentation du Pascal pour OpenVMS (http://h71000.www7.hp.com/commercial/pascal/documentation.html) (voir aussi ici (http://h71000.www7.hp.com/doc/pascal.html))
14. Documentation du Pascal de SGI (http://techpubs.sgi.com/library/tpl/cgi-bin/download.cgi?coll=0630&db=bks&docnumber=007-0740-030)
15. Documentation du Pascal sur CDC Cyber (http://bitsavers.trailing-edge.com/pdf/cdc/cyber/lang/pascal/)
16. Documentation du Pascal pour IBM 370 (http://bitsavers.trailing-edge.com/pdf/ibm/370/pascal/)
17. Documentation de Pascal-83 pour serveurs ClearPath d'Unisys (http://public.support.unisys.com/search/DocumentationSearch.aspx?ID=700&pla=ps&nav=ps)
18. Documentation du Pascal pour TI-990 (http://bitsavers.trailing-edge.com/pdf/ti/990/pascal/)
19. ISO 7185 (http://www.fh-jena.de/~kleine/history/languages/iso-iec-7185-1990-Pascal.pdf) sur le site de Karl Kleine
20. ISO 10206 (http://www.fh-jena.de/~kleine/history/languages/iso-iec-10206-1990-ExtendedPascal.pdf) sur le site de Karl Kleine
21. Use of the Overlay Technique in MS-DOS to Circumvent the 640K Conventional Memory Barrier (https://users.cs.jmu.edu/abzugcx/public/Student-Produced-Term-Projects/Operating-Systems-2002-FALL/MSDOS-by-Andrew-Vogan-2002-Fall.doc), Andrew C. Vogan, Cours CS 450, Automne 2002, université James Madison
22. Memories of Turbo Pascal version 1.0 - Anders Hejlsberg, United States (http://blogs.embarcadero.com/davidi/2008/11/20/39215), sur le blog de David Intersimone
23. Anders Hejlsberg (http://www.microsoft.com/about/technicalrecognition/anders-hejlsberg.aspx) Microsoft Technical Community Network - Awards and Recognitions

24. Guide de la programmation orientée objet dans Turbo Pascal 5.5 (http://edn.embarcadero.com/article/images/20803/TP_55_OOP_Guide.pdf)
25. Programmes (http://cm.bell-labs.com/cm/cs/who/bwk/pascaltools.txt) du livre *Software Tools* de Kernighan et Plauger
26. A collection of Pascal Programs (ftp://ftp.inf.ethz.ch/pub/publications/tech-reports/1xx/033.pdf) Niklaus Wirth, ETH Zurich, 1979
27. Why Pascal is not my Favourite Language (http://www.cs.bell-labs.com/cm/cs/cstr/100.ps.gz)
28. on peut considérer Modula comme une réponse au moins partielle à ces critiques
29. Proceedings (http://ntrs.nasa.gov/search.jsp?R=19790004544), Conference on the Programming Environment for Development of Numerical Software, 1978 (voir aussi ici (https://dx.doi.org/10.1145/1053417.806441))
30. ACM Special Interest Group on Numerical Mathematics (http://dl.acm.org/sig.cfm?id=SP944)
31. Présentation de Dennis Volper, membre de l'équipe développant le Pascal UCSD - p. 64 des Proceedings
32. John Reagan, Pascal Standards FAQ (http://pascal-central.com/extpascal.html)
33. Free Pascal Reference Guide, section 3.3.1 - Arrays (http://www.freepascal.org/docs-html/ref/refsu15.html)
34. Free Pascal Reference Guide, section 14.4.5 - Open array parameters (http://www.freepascal.org/docs-html/ref/refsu59.html)
35. Free Pascal Reference Guide, section 3.2.4 - Ansistrings (http://www.freepascal.org/docs-html/ref/refsu10.html)
36. Free Pascal Reference Guide, section 3.6 - Procedural types (http://www.freepascal.org/docs-html/ref/refse17.html)
37. ALESUP : Atelier logiciel de l'enseignement supérieur - Centre international de rencontres mathématiques (CIRM)
38. Jean-Louis Maltret et Robert Rolland, Mathématiques Algorithmique et Informatique (http://lumimath.univ-mrs.fr/~jlm/travaux/livretab/) Ellipses, 1994 (ISBN 2-7298-9410-1)
39. Patrick Cousot, Introduction à l'algorithmique numérique et à la programmation en Pascal, cours et exercices (http://www.di.ens.fr/~cousot/books/LT.shtml), McGraw-Hill, 1987 (ISBN 2-7042-1173-6)
40. Bruno Petazzoni, L'informatique dans les classes préparatoires aux grandes écoles (http://hal.archives-ouvertes.fr/docs/00/04/44/93/PDF/ba1p185.pdf), Revue de l'EPI (Enseignement Public et Informatique), n°101 (2001)
41. Denis Monasse, Point sur le programme de l'option informatique en classe MPSI (première année) (http://pauillac.inria.fr/~quercia/documents-info/Luminy-97/documents/sup.txt) compte rendu du débat de Luminy (1997)
42. Pierre Dieumegard, Comment appliquer l'algorithmique aux sciences expérimentales à partir de logiciels de mathématiques ? Quelques problèmes posés par le programme officiel de classe préparatoire BCPST (http://www.epi.asso.fr/revue/articles/a0809e.htm)
